International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Hybrid Cost Estimation Method for Planning Software Projects Using Fuzzy Logic and Machine Learning

Ajay Jaiswal*[1], Jagdish Raikwal[2], Pushpa Raikwal[3]

Submitted: 27/08/2023 **Revised**: 22/10/2023 **Accepted**: 01/11/2023

Abstract: Accurate cost prediction is crucial for efficient project management due to the potential for delays and budget over runs. The present research presents a novel method for accurately estimating the price of software projects by combining fuzzy logic with machine learning. The technique incorporates pre-processing procedures, feature selection, and fuzzy rule buildingto make use of historical data from datasets including "Desharnais," "Kitchenham," and "Maxwell." An ensemble classifier is built up out of several methods (such as Linear Regression (LR), Support Vector Machine (SVM), Feed Forward Neural Network (FNN), and Recurrent Neural Networks (RNN)) and then evaluated using various standards. Prominent results include the SVM model achieving the highest R-squared error in the Desharnais dataset, whereas the Ensemble model excelled in the Kitchenham dataset, achieving the highest R-squared error at 0.9307893, and the lowest Root Mean Squared Error at 0.2707119 among all models. In the Maxwell dataset, the LR model had the maximum R-squared error of 0.6073169, while the RNN model had the lowest R-squared error of 0.0237821. This method has the potential to improve software project planning in the actual world through accurate cost estimation.

Keywords: *Software, Cost, Estimation, Project, Machine Learning (ML), Fuzzy Logic.*

1. Introduction

The estimate of software development costs is a crucial step performed early in the process. The purpose of this method is to get a clearer picture of the project's development and objectives in the future. It is also important to have well-defined project parameters to help stakeholders manage the project's people, stuff, software, data, and even the feasibility study. The project manager's estimations of the project's cost, time, and resources or assets are greatly improved by accurate estimating findings [1]. However, errors in the project cost-estimating process might have a major impact on the timeliness and success of the project's completion. Inaccurate or incomplete project assessments could lead to delays or cost overruns in the delivery phase and problems with the quality and efficiency of the project's operations. As a result, software project cost estimate remains a challenging topic in the software engineering discipline [2]. To predict how much more accurately a project will cost, several models have been devised. Software cost estimates could be made with more precision if the right methods are used. In industrial economics, incorrect estimates of effort have been discovered to be quite hazardous logic. Cost estimation for software is a critical aspect of managing programs in production. The practice of anticipating software costs has advanced, although it is still imperfect. The ideal cost-estimation approach proposes a way to increase the value of the projected software cost by disclosing uncertainty 2 (in both comprehending the project and costing accuracy) and lowering the risk that the estimate will be far off from the real cost [3].

Fuzzy logic is a widely used method of estimate and prediction and is one kind of soft computing. Multiple decision-making programs now in use rely heavily on fuzzy logic [5]. Predictions using fuzzy logic have been used in a variety of fields, including medicine, politics, sports, the economy, etc. [6].

1.1 Cost Estimation Technique

The price of the software could be estimated in several ways [7]. One of the most important steps in creating new software is determining how much it will cost. This includes estimating the time frame, resources, and size of the project [8, 9]. It has been estimated that software project estimates might be off by as much as 40% [10]. These methods might be broken down into two broad categories: algorithmic and non-algorithmic approaches. In this part, the author details various methods and their relative merits and demerits so that one can make an informed decision about which approach is best [11]. Estimating how much a software project will cost is a crucial part of software engineering, one that may make or break a deal or a project. During the software development life cycle, determining how much work will be needed and how much it will cost is the primary focus of software cost and effort estimation [12].

[1] *Prestige Institute of Engineering Management and Research, Indore (MP) 4520007, India.*
ORCID ID: 0009-0002-3313-0912
[2] *IET, DAVV, Indore (MP), 452001, India.*
[3] *PDPM-IIITDM, Jabalpur, 482005 India.*
** Corresponding Author Email: ajay.jaiswal55555@gmail.com, jraikwal@ietdavv.edu.in, praikwal@iiitdmj.ac.in*

1.1.1 Non-algorithmic Techniques

The estimating procedure of non-algorithmic methods is based on analogy and deduction. The team working on the software project needs background information on a comparable project that was completed in the past. Past software projects or data sets are examined to provide a foundation for estimates [13]. Some of the Non-Algorithmic approaches are described in more depth below:

Expert Judgement: Traditional methods of software cost assessment sometimes include the use of expert opinion throughout the planning and prototyping stages of software development. Due to the nature of the method, a professional cost estimator's knowledge and experience are crucial to its success [14]. The expert's subject expertise determines it. Because of their expertise, skilled estimators estimate software costs. EJ is a method for estimating software project costs by consulting with experts. EJ relies on expert experience, expertise, motives, the area's knowledge, and analyst-expert discussions. According to Cooke, EJ's most important instrument is reluctance. Even though non-cost estimators generally frown upon the usage of EJ in a parallel engineering setting, it is becoming increasingly commonplace [15]. Despite the lack of study on statistical gathering techniques, there is a widespread belief that the procedure must be mostly spontaneous and hence unfair and sensitive to political pressures [16].

Estimation Techniques: Expert-based cost estimates reflect the expertise of the experts who were consulted and are hence reliant on the projects in which they were used. Data collection and discovery could be hampered in several everyday situations. These are cases when the "expert judgment" approach works well [17]. It is the standard method of calculating how long a software project will take. The Wideband Delphi Method is one of the techniques used to estimate costs using expert opinion. There are two rounds of evaluation for these individuals. One other instance of expert opinion is the work breakdown structure [18].

Top-Down Estimating Method: The term "Macro Model" is often used to refer to the top-down estimation technique it describes. Using this technique, the overall software project cost estimate is determined from the project's global attributes, and then the project is broken down into its constituent low-level mechanisms or components. The Putnam model is a technique that takes this perspective. For preliminary cost calculation when only global parameters are available, the Top-Down approach is preferable. Due to a lack of specifics at the outset, top-down approaches are ideal for estimating software costs.

Bottom-up Estimating Method: A predicted total project cost is then calculated by adding the individual product costs determined using the base-up costing method. The goal of a bottom-up approach is to build a framework's gauge from data gathered about its constituent parts and how they interact. The point-by-point model used by COCOMO is the technique using this approach [19].

1.1.2 Algorithmic Techniques

The goal of these techniques is to help with the calculation of software costs by providing some mathematical formulae. Inputs such as Source Lines of Code, the number of required functions, and cost drivers such as language, design approach, skill levels, risk assessments, etc. are used as the basis for these mathematical calculations, which are based on research and historical data [20]. The application of Algorithms created a plethora of models, including COCOMO, Putnam, and function point models.

Constructive Cost Model (COCOMO): A new software development activity's time, money, and effort costs might be estimated using the COCOMO model. Lines of code are used as the dependent variable in the COCOMO model, which is a regression model. It is based on research into a variety of past projects' worth of data [21]. The three levels of COCOMO that Boehm proposed are Basic, Intermediate, and Detailed. The basic COCOMO project looked at how much work is related to program size [22]. The model's accuracy is low since it ignores the effect of several important variables. It was suggested that Intermediate COCOMO be made better by considering a group of 15 cost drivers depending on different aspects of software development. At each stage/module of the software development process, the comprehensive COCOMO evaluates all the cost-contributing elements listed in Intermediate COCOMO [23].

Putnam model: This model is based on the analysis and discovery of several completed projects, as well as the distribution of labor suggested by Rayleigh and Norden. The software equation, which is the central component of Putnam's model, is defined as:

$$S = E * Efforts^{1/3} * t_d^{4/3} \quad (1)$$

Here, td is the software delivery time, and E is the environmental element that duplicates the development skills that can be extracted from past data using software equations. The effort is measured in CY, while the magnitude of S is expressed in the line of code (LOC). Another fundamental connection first identified by Putnam is.

$$Effort = D_0 * t_d^3 \quad (2)$$

Here, D_0 is a parameter for the size of the workforce, which could be anywhere from 8 (for brand-new software) to 27 (for heavily modified programs) [24]. Putnam's model is often used in SLIM and for prepping. SLIM is a resource for workforce planning and estimating that is

based on Putnam's model [25].

Function Point-Based Analysis: Function point analysis (FPA) is a popular method for assessing software cost-estimating frameworks. Each function is assessed by its functional complexity throughout the point-counting procedure that estimates an undertaking or application. Software must meet FPA requirements. These functions relate to information software uses and generation [26]. It compares a software's fourteen general system characteristics (GSCs) data input from external input, external inquiry, external output, and internal logical file (that is, the functionality needed by and provided to the end consumer) against the standard standards [27].

2. Literature Review

This section examines the literature on a Hybrid Cost Estimation Method for Planning Software Projects using Fuzzy Logic and Machine Learning, discussing, and analyzing the relevant work by different authors.

Butt et al., (2022) [28] provide a framework to deal with these variables and prevent unnecessary delays or extra expenses. The author used estimating methods in a variety of software industry case studies, with similar outcomes. The method calculates efforts independently. The author provides a software-based estimating method that takes team input and predicts project cost and time. The author observed that the cost-estimating technique minimized agile software project development difficulties and improved project estimation. The suggested estimate method provided a novel notion of estimation that helps clients, the software industry, and programmers all work together to satisfy clients, accommodate changes made during sprints, and complete projects on schedule and under budget.

Dashti et al., (2022) [29] examined how the Learnable Evolution Model (LEM) algorithm affected feature weighting optimization and presented an alternative approach. The performance of this approach is evaluated in this study using the Desharnais and Maxwell datasets. The suggested technique has been tested and compared to various evolutionary algorithms using the mean magnitude of Relative Error (MMRE), Percentage of Predictions (PRED (0.25)), and Median Magnitude of the Relative Error (MdMRE) criteria. When applied to software cost estimates, the suggested technique shows significant improvement. The suggested model outperforms the attribute-based encryption (ABE) model across all three metrics (MMRE, MdMRE, and PRED (0.25) with an average improvement of 86%, 79%, and 64%, respectively; however, this does not reveal the breadth of the test set.

Abdulmajeed et al., (2021) [30] use a new model based on machine learning to determine how much a software engineering project will cost. The Elman Neural Network (ENN), Cascade Neural Network (CNN), and K-Nearest Neighbors (KNN) algorithms were used to forecast the expenses associated with creating software projects. NASA project data was used to evaluate the models, and the results were compared to those obtained using KNN, CNN, and ENN. As shown by the findings, the KNN was the most effective method for cost estimation and prediction, while the based model achieved the best results in terms of MMRE, RMSE, and Balanced Relative Error (BRE) (0.101, 0.547, and 0.205, respectively). The suggested approach was shown to be accurate to the tune of 90.238% when the necessary calculations were made. This indicates that the system is very accurate and displays little variance using the KNN method for cost prediction.

Karimi and Gandomani (2021) [31] Introduce a novel model built on a combination of ANFIS (Adaptive Network-Based Fuzzy Inference System) and DE (Differential Evolution). The goal of this model is to improve upon earlier research by providing a more precise estimate of the time and effort required to build software. The suggested technique improved accuracy by up to 7% as measured by the MMRE and Prediction (PRED) (0.25) criterion, outperforming existing optimization algorithms adapted from the genetic algorithm, evolutionary algorithm, meta-heuristic algorithm, and neuro-fuzzy-based optimization algorithm.

Priya et al., (2021) [32] provided a method for estimating software development time utilizing an array of methods, including machine learning and deep neural networks. Ensemble methods involve combining numerous independent models. The estimate strategies of averaging, weighted averaging, bagging, boosting, and stacking were classified as ensemble approaches. The generalized linear model, the decision tree, the support vector machine, and the random forest were among the stacking models explored and compared. For the Albrecht, China, Desharnais, Kemerer, Maxwell, Kitchenham, and Cocomo81 datasets, the RF model stacking presented yielded values of 0.9357274, 0.9839643, 0.65561700, 0.7520435, 0.8120214, 0.9246614, and 0.8667750.

Ullah et al., (2020) [33] provide a contrast between two algorithmic approaches, the Baily-Base d model, and Constructive Cost Models (COCOMO-II). Both Turkish and NASA data sets are used in the simulation. The following findings demonstrate that COCOMO-II outperforms Bailey Basili in MMRE (Mean Magnitude of Relative Error).

Qiao et al., (2020) [34] suggest a new method for estimating software flaws, one that makes use of deep learning. The evaluation findings show that the suggested method is reliable and can outperform existing methods. The mean square error is reduced by over 14% on average,

and the squared correlation coefficient is increased by over 8% when using the proposed strategy.

Asheeri et al., (2019) [35] built a machine learning-based software price prediction model. To make an early-stage software cost prediction, a variety of machine learning methods are used on two publicly available datasets. In this study, the author employed the Root Relative Squared Error (RRSE), the Relative Absolute Error (RAE), the Mean Absolute Error (MAE), and the square root of the RMAE as assessment metrics. The suggested model's goal is to try prediction using the features of the dataset, then to compare the predicted and real efforts and evaluate the inaccuracy using various metrics. In this case, a higher R2 value indicates a better outcome, but lower values indicate a better result for the other metrics. The results demonstrate the efficacy of using machine learning techniques for software price forecasting.

Mojeed et al., (2019) [36] introduce a memetic method to the issue of calculating when programmers should work extra. Overtime hours, project completion time, and overall budget are the three objectives of this optimization issue. The issue is solved by using the Multi-Objective Shuffled Frog-Leaping Algorithm (MOSFLA), an algorithm developed for overtime scheduling. The author employed three popular multi-objective quality indicators to conduct empirical assessment studies on six real-world software project datasets. Contribution (IC) values of 0.0118, Hyper volume (IHV) values of 0.0102, and Generational Distance (IGD) values of 0.0102 were achieved by MOSFLA, showing that it considerably exceeded the current typical overtime management solutions in software engineering projects.

Ullah et al., (2019) [37] suggest utilizing a typical Turkis industry dataset in conjunction with a Flower Pollination Algorithm (FPA) to optimize the parameters of a Constructive Cost Model II (COCOMO-II). According to the experimental results, the suggested method provides a more accurate estimation than the Bat algorithm and the original COCOMO-II in terms of the Manhattan distance (MD) and the mean magnitude of relative errors (MMRE).

Jha et al., (2019) [38] apply deep learning to forecast software maintainability indicators using several data sets. Offers indicators that may be utilized for software maintenance forecasting, and the suggested deep learning model is more effective than any of the other approaches studied. Furthermore, the experimental outcomes validate the effectiveness of the proposed deep learning model for software maintainability prediction.

Saljoughinejad et al., (2018) [39] purpose to boost the COCOMO model's precision in cost estimation. We use meta-heuristic methods to examine the key factors in costs. In this approach, COCOMO is enhanced in a distinguishable fashion through careful selection of coefficients and reconstruction. Using data from actual software development projects, researchers found that the IWO-PSO hybrid model yielded the most precise outcomes.

Puspaningrum (2017) [40] presented a combination of the cuckoo search and the harmony search method to fine-tune the values of the four COCOMO-II parameters for maximum accuracy. The suggested method is tested on the NASA 93 dataset and analyzed with the Mean Absolute Error (MAE) and the Mean Squared Error (MSSE). The suggested technique outperforms COCOMO-II and the cuckoo search algorithm in estimating the time and effort required to complete a software project, as shown by the experiments.

Miandoab et al., (2016) [41] proposed a hybrid approach that is utilized, which takes the best features of the COA-Cuckoo optimization process and the K-Nearest Neighbors (KNN) algorithm. Composition algorithms are tested on six distinct datasets using eight distinct metrics. Estimated costs were shown to be more accurate because of these analyses. Based on the findings, the typical relative error for the COCOMO model is the corresponding numbers for the genetic algorithm and the firefly method are 38.31 and 30.34, respectively. In the compositional model, this value is equal to 22.53.

Yadav et al., (2015) [42] implement the most important criteria for dependability at each stage of the software development life cycle (SDLC), and a fuzzy logic-based phase-wise software defect prediction model is provided. Defect density indication is forecasted across the phases of requirement analysis, design, coding, and testing with the use of nine software metrics. Twenty sets of data from actual software projects are used to verify the proposed model's prediction accuracy. There are no major issues with the validation. The average relative error and the balanced mean relative error are two measures that tend to improve with software project size.

Gharehchopogh et al., (2015) [43] investigated SCE by combining the Meta-Heuristic Search Techniques of the Genetic Algorithm (GA) and the Artificial Bee Colony (ABC). The MRE error levels for the suggested model, GA, and ABC algorithms are all lower than those of the COCOMO model, as shown by the test results. The convergence of the hybrid model is superior to that of the GA and ABC algorithms.

Dizaji et al., (2015) [44] suggested using meta-heuristic methods to calculate software project costs. In this work, we apply the Ant Colony Optimization (ACO) and Lorentz transformation as a Chaos Optimization Algorithm (COA) using datasets from NASA for both development and evaluation. The suggested technique is compared and

evaluated with the COCOMO model using MARE, and the findings reveal a decrease in MARE to 0.078%.

Du et al., (2015) [45] estimate the software costs, and a neuro-fuzzy model has been developed, which combines a neural network model with a fuzzy model. According to the obtained data, the suggested model can enhance the estimation accuracy by 18% using the Mean Magnitude of Relative Error (MMRE) criteria.

Shepperd et al., (2014) [46] attempt to decipher the seemingly contradictory experimental findings and identify the variables most responsible for variation in predicting accuracy. To identify the characteristics that affect predictive ability, authors undertake a meta-analysis of all applicable, high-quality primary studies of defect prediction. Surprisingly, authors discover that the classifier selection has a minor effect on performance (1.3%), whereas the largest (31%). Researchers themselves serve as an explanatory variable. Who does a task is more important than the task itself.

Attarzadeh et al., (2010) [47] aim to increase precision in software project estimates by putting out a novel realistic model based on fuzzy logic. The primary goal of this study was to examine how the fuzzy logic method, by describing input parameters with a two-sided Gaussian function that provided a superior transition from one interval to another, may enhance the precision with which effort estimates are made. The results obtained by applying COCOMO II and the proposed model based on fuzzy logic to the NASA dataset and the artificial dataset created for comparison revealed that the proposed model was performing better than ordinary COCOMO II and that the achieved results were closer to the actual effort.

Attarzadeh et al., (2010) [48] introduce a novel fuzzy logic model to address the estimated inaccuracy introduced by the COCOMO model's dependence on input data. The primary goals of this study are to examine how applying fuzzy logic to the model results in more precise software effort estimations and to identify the impact of crisp inputs and fuzzification approaches on the accuracy of the system's output. The empirical results indicated that the MMRE and the Pred (0.25) for projects with a relative error of less than or equal to 0.25 were both marginally reduced when using fuzzy logic for software effort estimations.

3. Proposed Methodology

This section shows the working principle of A Hybrid Cost Estimation Method for Planning Software Projects using Fuzzy Logic and Machine Learning [49]. Below is the step-by-step description of the proposed methodology and Fig 1 shows the flow diagram.

The suggested approach for estimating the cost of software projects makes use of datasets with the names "Desharnais", “Kitchenham", and "Maxwell". These statistics provide historical data about software projects. Some of the topics included in this data are project size and complexity, as well as team experience and real expenses. To get the data ready for use, several pre-processing activities such as addressing missing values and scaling features are carried out. To determine which characteristics should be extracted for use in cost estimating, the author uses the TF-IDF. Lasso is used for feature selection to zero in on relevant characteristics. The terms "low", "medium" and "high" for project size are examples of the linguistic variables used in the development of fuzzy rules, which are in turn based on expert knowledge and historical data. The AND, OR, and NOT operators are used to combine these fuzzy rules to form fuzzy inference rules.

Then, the defuzzification process is used on the hazy output memberships to provide precise numbers for use in the cost-estimating process. The suggested method makes use of a multi-model ensemble classifier that draws from LR, SVM, Feed FNN, and RNN techniques. An ensemble classifier is created by combining the results of many individual classifiers using fuzzy logic. The performance of the ensemble classifier is measured in terms of metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) [50], and compared to other classifiers like Random Forest (RF). When it comes to real-world software project planning, the optimum cost-estimating model based on fuzzy logic is used. Before being used for real-world cost prediction in software projects, the final ensemble classifier is verified on fresh, unseen software project data to guarantee its generalizability and reliability.

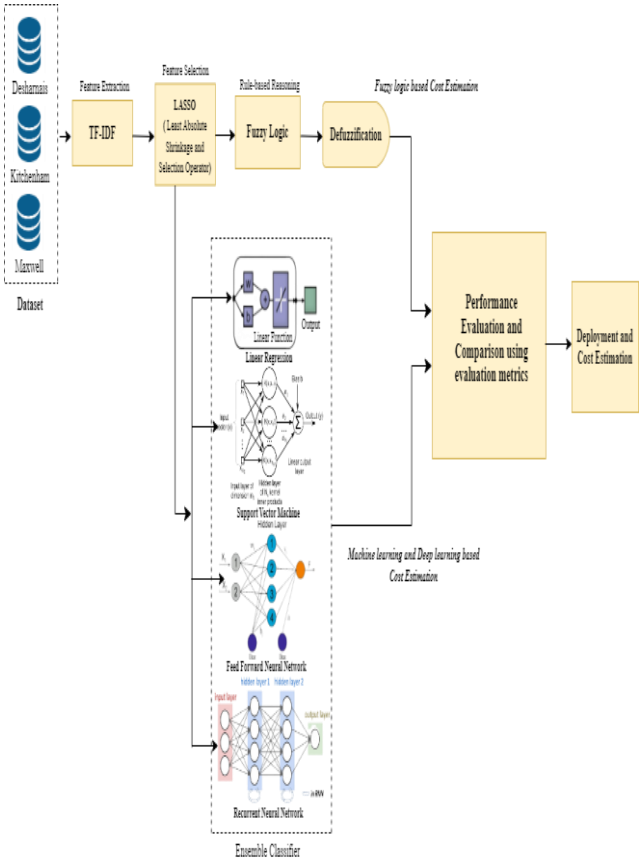

Fig.1. Architectural Structure of Methodology

4. Proposed Algorithm

Algorithm: Pseudocode of proposed Cost Estimation Method for Planning Software Projects using Fuzzy Logic and Machine Learning

Data Preprocessing

```
def preprocess_data(dataset):
preprocessed_datasets = preprocess(datasets)
return preprocessed_datasets
```

Feature Extraction:

```
def extract_features_with_tfidf(dataset):
tfidf_vectorizer = TfidfVectorizer()
tfidf_features = tfidf_vectorizer.fit_transform(dataset)
return tfidf_features
```

Feature Selection:

```
def select_features_with_Lasso(dataset):
selected_features = select_features(dataset)
lasso_model.fit(dataset, selected_features)
return selected_features
```

Fuzzy Logic Rule Development

```
def develop_fuzzy_rules(dataset):
fuzzy_rules = create_fuzzy_rules(dataset)
return fuzzy_rules
```

Defuzzification

```
def defuzzify_output(fuzzy_output):
crisp_output = defuzzification_process(fuzzy_output)
return crisp_output.
```

Ensemble Classifier Construction

```
def train_base_classifiers(dataset):
base_classifier_lr = train_linear_regression(dataset)
base_classifier_svm =
    train_support_vector_machine(dataset)
base_classifier_fnn =
    train_feed_forward_neural_network(dataset)
base_classifier_rnn = train_re-
    current_neural_network(dataset)
return base_classifier_lr, base_classifier_svm,
    base_classifier_fnn, base_classifier_rnn
```

Fuzzy Logic-based Ensemble Approach

```
def fuzzy_logic_ensemble(base_outputs):
  ensemble_output =
    combine_outputs_with_fuzzy_logic(base_outputs)
  return ensemble_output
```

Performance Evaluation and Comparison

```
def evaluate_classifier_performance(ensemble_output,
    actual_costs):
  rmse = root_mean_squared_error(ensemble_output,
    actual_costs)
def compare_performance_with_rf(ensemble_output,
    actual_costs):
  rf_output = random_forest_classifier(dataset)
   return rf_rmse, fuzzy_logic_rmse
```

Cost Estimation

```
estimated_costs =
    optimized_fuzzy_logic_classifier.predict(new_data)
  return estimated_costs.
```

5. Result and Discussion

This section provides a comprehensive and in-depth examination of the results and discoveries that resulted from the proposed methodology. The discussion incorporates numerous crucial aspects, such as feature extraction, feature selection, fuzzy logic rule formulation

procedures, and performance evaluation. This analysis highlights not only the successes of the hybrid method for estimating software project costs but also its prospective implications in actual planning scenarios. The results and discussion section provides an in-depth summary of the proposed Hybrid Cost Estimation Method for Planning Software Projects utilizing Fuzzy Logic and Machine Learning. This section describes the methodology's essential stages and offers insight into the performance of the proposed approach.

5.1 Feature Extraction and Selection for Cost Estimation

The process of extracting features is a crucial step in the technique that has been developed for predicting the costs of software projects. In this part of the analysis, the method of TF-IDF is used to extract relevant characteristics from the datasets, namely Desharnais, Kitchenham, and Maxwell, to add these characteristics into the procedure of cost estimate. The TF-IDF technique is used to extract features from the Desharnais, Kitchenham, and Maxwell datasets, and those features are shown in Table 1 below.

Table 1. Extracted Features using TF-IDF

Dataset	Extracted Features
Desharnais	Project, TeamExp, ManagerExp, YearEnd, Length Transactions, PointsNonAdjust, Adjustment, PointsAjust, Effort
Maxwell	Year, App, Har, T14, T06, Source, Nlan, T05, T09, T15, Duration, Size, Time, Effort
Kitchenham	Clientcode, Projecttype, Duration, Adjfp, Estimate, Estimate method, Effort

Using the Lasso regression method, Table 2 provided a summary of selected features gathered from various datasets. Lasso is a technique used in regression analysis for variable selection and regularization to prevent overfitting. Each entry in the table represents a unique dataset, and the columns provide information about the designated features for each dataset. Here is an outline of the table:

- **Dataset:** This column identifies the dataset for which feature selection was conducted. There are three mentioned datasets: Kitchenham, Maxwell, and Desharnais.
- **Selected Features:** This column lists the features (variables or attributes) selected using the Lasso feature selection method for each dataset. These characteristics are the most pertinent or instructive for the analysis or modeling duties related to each dataset.

Table 2. Selected Features using Lasso.

Dataset	Selected Features
Desharnais	Project, TeamExp, Transactions, Points NonAdjust, PointsAjust, Effort
Maxwell	App, Har, Year, Source, Nlan, T05, T09, T15, Duration, Size, Time, Effort
Kitchenham	Clientcode, Project type, Duration, Adjfp, Estimate, Effort

5.2 Key Fuzzy Rules for Cost Estimation and Effort Prediction

Fuzzy rules serve as a set of guidelines for predicting "cost estimation" in the "Desharnais" dataset and "effort" in the "Kitchenham" and "Maxwell" datasets, respectively. These principles are intended to provide qualitative and approximate predictions, making them particularly useful in situations where the relationships between variables are not precisely defined.

The provided fuzzy rules establish relationships between distinct input variables and the resulting output variable, "Effort," for three distinct datasets: "Desharnais," "Kitchenham," and "Maxwell." These rules are an integral part of fuzzy logic systems, allowing for the formation of well-informed decisions and forecasts in the face of ambiguous or imprecise information. Tables 3, 4, and 5 contain these fuzzy rules, which play a crucial role in utilizing fuzzy logic to manage situations characterized by ambiguous or imprecise information.

Table 3. The fuzzy rule for the “Desharnais” dataset

No.	Rule
1.	IF TeamExp is Low AND ManagerExp is Low THEN CostEstimation is High
2.	IF TeamExp is Low AND ManagerExp is Medium THEN CostEstimation is Medium
3.	IF TeamExp is Low AND ManagerExp is High THEN CostEstimation is Medium
4.	IF TeamExp is Medium AND ManagerExp is Low THEN CostEstimation is Medium
5.	IF TeamExp is Medium AND ManagerExp is Medium THEN CostEstimation is Medium
6.	IF TeamExp is Medium AND ManagerExp is High THEN CostEstimation is Low
7.	IF TeamExp is High AND ManagerExp is Low THEN CostEstimation is Low
8.	IF TeamExp is High AND ManagerExp is Medium

	THEN CostEstimation is Low
9.	IF TeamExp is High AND ManagerExp is High THEN CostEstimation is Low

Table 4. The fuzzy rule for the "Kitchenham" dataset

No.	Rule
1.	IF Project Type is 'A' OR 'D' AND Duration is long AND Estimate method is 'A' OR 'CAE' THEN Effort is High.
2.	IF Project Type is 'P' AND Adjusted function points are large AND Estimate method is 'C' OR 'EO' THEN Effort is Very High.
3.	If the Project Type is 'U' Duration is short AND the Estimate method is 'W' THEN Effort is Low.
4.	IF Client code is 1 AND Adjusted function points are small AND Estimate method is 'EO' THEN Effort is Low.
5.	IF Client code is 6 AND Duration is medium AND Estimate method is 'A' THEN Effort is Medium

Table 5. The fuzzy rule for the "Maxwell" dataset

No.	Rule
1.	If App is InfServ and Har is PC Dba is GUI and UI_Source is Outsrced, then Effort is Low.
2.	If App is TransPro and Har is Mainfrm Dba is TextUI and UI_Source is Inhouse, then Effort is Medium.
3.	If App is CustServ and Har is Multi Dba is TextUI and UI_Source is Outsrced, then Effort is High.
4.	If App is MIS Har is Network Dba is GUI and UI_Source is Inhouse, then Effort is Very High.
5.	If Telonuse is yes and Nlan is 1, then Effort is Low.
6.	If Telonuse is no and Nlan is 4, then Effort is High.
7.	If T01 is between 1 and 3, then Effort is Low.
8.	If T01 is between 2 and 4, then Effort is Medium.
9.	If T01 is between 3 and 5, then Effort is High.
10.	If T02 is between 1 and 2, then Effort is Low.

These fuzzy rules are essentially a set of conditional statements that make predictions or classifications using linguistic variables (e.g., low, medium, high). They are derived from databases using an inference system based on fuzzy logic. The principles can be used to estimate costs or efforts based on specific project attributes or characteristics.

5.3 Performance Evaluation

Table 6 displays the performance metrics of various machine learning models (LR, SVM, FNN, RNN, and Ensemble) on the Desharnais dataset. The table contains different kinds of errors (R-squared error, Root Mean Squared Error, and Mean Absolute Error) calculated for each model.

- **R-squared error (RE)** is a statistical measure that indicates the proportion of the variance in the dependent variable that can be predicted by the independent variables.
- **Root Mean Squared Error (RMSE)** quantifies the average difference between predicted and actual values. It indicates the typical magnitude of error by giving greater weight to significant errors.
- **Mean Absolute Error (MAE)** is another method for calculating the average deviation between predicted and actual values. Contrary to RMSE, it considers all errors equally, irrespective of their magnitude.

There are corresponding values for each error metric for each model (LR, SVM, FNN, RNN, and Ensemble) in the Desharnais dataset. These values indicate how well each model performed relative to the error metric in question.

Table 6. Desharnais dataset

Errors	LR	SVM	FNN	RNN	Ensemble
RE	0.7776860	0.8037751	0.7368849	0.6715072	0.7981238
RMSE	0.3528396	0.2221946	0.2932015	0.3319024	0.2411772
MAE	0.2628982	0.3314904	0.3838547	0.4289008	0.3362300

Fig 2 is a graphical representation of performance metrics for several machine learning models (LR, SVM, FNN, RNN, and Ensemble) applied to the Desharnais dataset. R-squared error, Root Mean Squared Error, and Mean Absolute Error were computed for each model and are depicted in the figure.

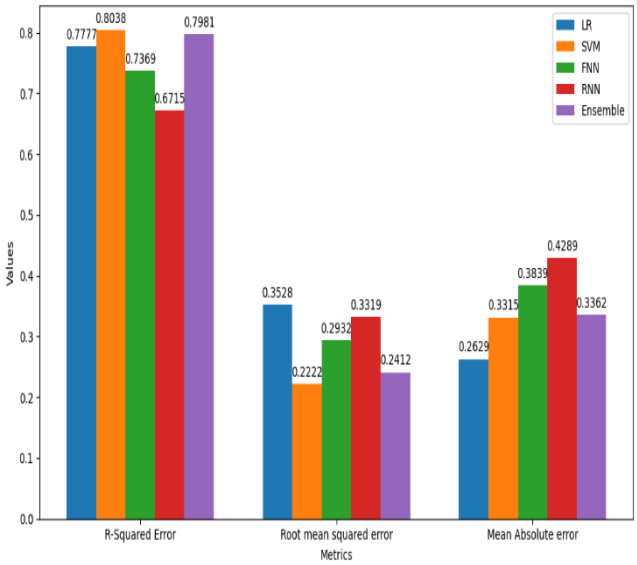

Fig.2. Performance metrics comparison

Error metrics for multiple machine learning models on the Maxwell dataset appear to be summarized in Table 7.

Table 7. Maxwell Dataset

Errors	LR	SVM	FNN	RNN	Ensemble
RE	0.6073169	0.4469283	0.2395082	0.0237821	0.5078570
RMSE	0.5881401	0.5372502	0.6687865	0.6857223	0.5213621
MAE	0.4839028	0.6979923	0.8184787	0.9273279	0.6584238

Error values for multiple machine learning models on the Maxwell dataset are graphically shown in Fig 3. Error metrics for many models are compared to provide light on how effectively each performs on the dataset, as seen in the image.

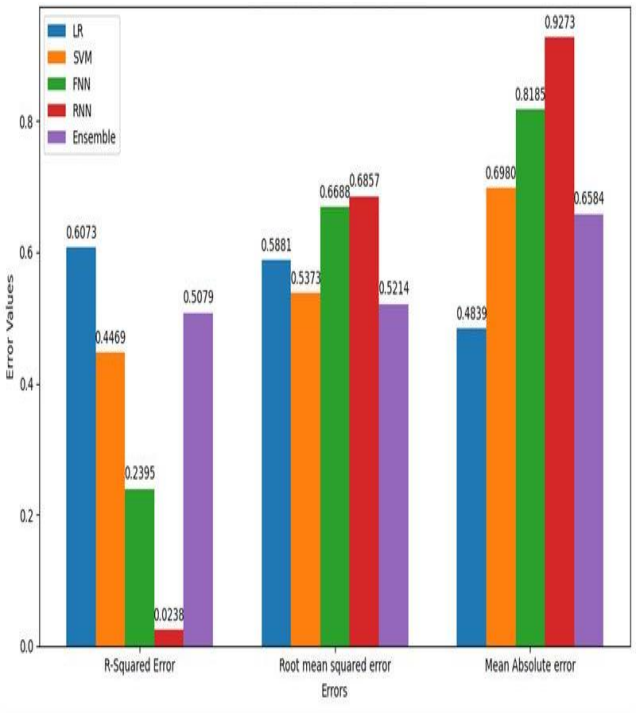

Fig.3. Error Values for Different Models

Metrics for each model's performance on R-squared error, root mean squared error, and mean absolute error are shown in Table 8. The table below displays the results of several models in terms of various error metrics. Error metrics for each model on the Kitchenham dataset are shown in the cells.

Table 8. Kitchenham dataset

Errors	LR	SVM	FNN	RNN	Ensemble
RE	0.9287298	0.9288793	0.9100070	0.8583288	0.9307893
RMSE	0.2747101	0.2744218	0.3086917	0.3873126	0.2707119
MAE	0.2009207	0.2021575	0.2493339	0.3067252	0.1941534

Error scores for several machine learning models applied to the Kitchenham dataset are graphically shown in Fig 4. This visual representation of error metrics allows for quick comparison between models, yielding useful insights into how they perform on the dataset.

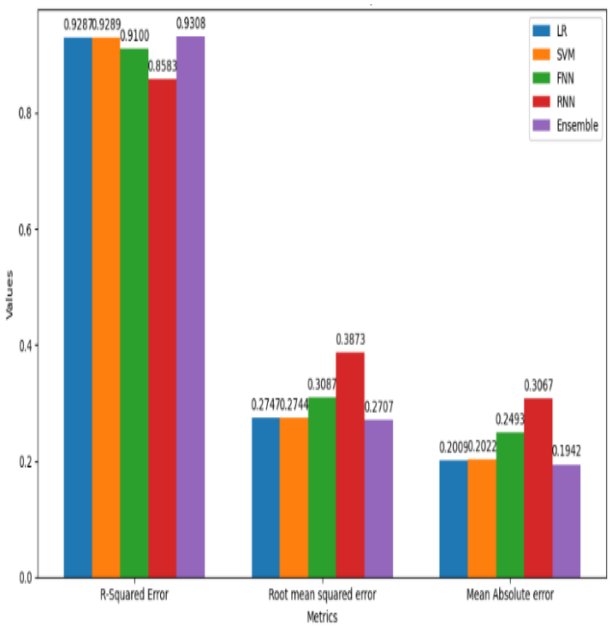

Fig.4 Performance Metrics Comparison

6. Conclusion

The paper presents a hybrid approach, incorporating fuzzy logic and machine learning, for precise cost estimation of software projects. Effective project management requires accurate cost estimation. Inaccuracy can cause delays and cost increases. This paper examines both algorithmic and non-algorithmic techniques for cost estimation, highlighting their advantages and disadvantages. The paper also provides a literature review on this hybrid approach, with an emphasis on its potential impact on software project planning. The suggested approach to cost estimate employs a hybrid of fuzzy logic and machine learning, drawing on the results of previous calculations performed on datasets such as "Desharnais," "Kitchenham," and "Maxwell." An ensemble classifier based on LR, SVM, FNN, and RNN is developed, and the author details the pre-processing stages, feature selection, and fuzzy rule generation that led to this outcome. This model is evaluated using multiple indicators to determine its relevance to real-world software project planning. The result and discussion section details the fuzzy logic and machine learning software project cost estimating approach. It covers essentials including feature extraction, selection, fuzzy logic rule development, and performance evaluation. Datasets like Desharnais, Kitchenham, and Maxwell are used to show how extracted attributes affect cost estimation. The section emphasizes fuzzy rules' role in approximation and qualitative predictions from unclear data. On each dataset, R-squared error, Root Mean Squared Error, and Mean Absolute Error are used to evaluate machine learning models (LR, SVM, FNN, RNN, Ensemble). These indicators show the models' performance and applicability for real-world software project planning. The results show that the SVM model achieved the maximum R-squared error of 0.8037751 in the Desharnais dataset, while the RNN model obtained the minimum at 0.67150729. For the Maxwell dataset, the LR model exhibited the highest R-squared error of 0.6073169, while the RNN model achieved the lowest value at 0.0237821. Finally, for the Mean Absolute Error, the RNN model exhibited the maximum value of 0.3067252 in the Kitchenham dataset, while the Ensemble model demonstrated the minimum at 0.1941534. The Ensemble model achieved a maximum R-squared error of 0.9307893 in the Kitchenham dataset and had a Root Mean Squared Error of 0.2707119, which was the lowest among all models. An improvement in machine learning algorithms and computational capacity may result in cost estimation models that are even more accurate and efficient. This hybrid approach has the potential to have a substantial impact on software project planning, making it an intriguing area for future research and development.

Conflicts of interest

The author(s) declared no potential conflicts of interest concerning this article's research, authorship, and publication.

References

[1] R. R. Sinha and R. K. Gora, "Software effort estimation using machine learning techniques," Advances in Information Communication Technology and Computing: Proceedings of AICTC 2019, pp. 65-79, 2021.

[2] S. Kumari and S. Pushkar, "Cuckoo search based hybrid models for improving the accuracy of software effort estimation," Microsystem Technologies, vol. 24, no. 12, pp. 4767-4774, 2018.

[3] R. Aziz, "Development of Simple Effort Estimation Model based on Fuzzy Logic using Bayesian Networks."

[4] S. M. R. Chirra and H. Reza, "A survey on software cost estimation techniques," Journal of Software Engineering and Applications, vol. 12, no. 6, p. 226, 2019.

[5] I. Maleki, L. Ebrahimi, S. Jodati, and I. Ramesh, "Analysis of software cost estimation using fuzzy logic," International Journal in Foundations of Computer Science & Technology (IJFCST), vol. 4, no. 3, pp. 27-41, 2014.

[6] R. P. S. Bedi and A. Singh, "Software Cost Estimation using Fuzzy Logic," Indian Journal of Science and Technology, vol. 10, p. 3, 2017.

[7] P. Pandey, "Analysis of the techniques for software cost estimation," in 2013 Third International Conference on Advanced Computing and

Communication Technologies (ACCT), 2013: IEEE, pp. 16-19.

[8] P. Sharma and J. Singh, "Systematic literature review on software effort estimation using machine learning approaches," in 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS), 2017: IEEE, pp. 43-47.

[9] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," Information and Software Technology, vol. 54, no. 1, pp. 41-59, 2012.

[10] K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. Hove, "A survey on software estimation in the Norwegian industry," in 10th International Symposium on Software Metrics, 2004. Proceedings., 2004: IEEE, pp. 208-219.

[11] S. Shekhar and U. Kumar, "Review of various software cost estimation techniques," International Journal of Computer Applications, vol. 141, no. 11, pp. 31-34, 2016.

[12] R. Tripathi and P. Rai, "Comparative study of software cost estimation technique," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 1, 2016.

[13] N. Balaji, N. Shivakumar, and V. V. Ananth, "Software cost estimation using function point with non-algorithmic approach," Global Journal of Computer Science and Technology Software & Data Engineering, vol. 13, no. 8, pp. 1-4, 2013.

[14] Z. B. Mansor, Z. M. Kasirun, N. H. H. Arshad, and S. Yahya, "E-cost estimation using expert judgment and COCOMO II," in 2010 International Symposium on Information Technology, 2010, vol. 3: IEEE, pp. 1262-1267.

[15] C. Rush and R. Roy, "Expert judgement in cost estimating: Modelling the reasoning process," Concurrent Engineering, vol. 9, no. 4, pp. 271-284, 2001.

[16] B. Khan, W. Khan, M. Arshad, and N. Jan, "Software cost estimation: Algorithmic and non-algorithmic approaches," Int. J. Data Sci. Adv. Anal, vol. 2, no. 2, pp. 1-5, 2020.

[17] A. Saeed, W. H. Butt, F. Kazmi, and M. Arif, "Survey of software development effort estimation techniques," in Proceedings of the 2018 7th International Conference on software and computer applications, 2018, pp. 82-86.

[18] P. P. Win, W. W. Myint, H. P. P. Mon, and W. Thu, "Review on Algorithmic and Non-Algorithmic Software Cost Estimation Techniques," International Journal of Trend in Scientific Research and Development (ijtsrd), pp. 890-895, 2019.

[19] K. Shweta, S. Duraismy, and T. LathaMaheswari, "Comparative Analysis of Algorithmic, Non Algorithmic and Machine Learning Models For Software Cost Estimation: A Survey," IRJMETS ISSN, pp. 2515-8260.

[20] L. Nerkar and P. Yawalkar, "Software cost estimation using algorithmic model and non-algorithmic model a review," Int J Comput App, vol. 2, pp. 4-7, 2014.

[21] A. Trendowicz and R. Jeffery, "Software project effort estimation," Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags, vol. 12, pp. 277-293, 2014.

[22] P. Brar and D. Nandal, "A Systematic Literature Review of Machine Learning Techniques for Software Effort Estimation Models," in 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT), 2022: IEEE, pp. 494-499.

[23] U. K. Christopher and A. C. Izuchukwu, "Trends and Techniques of Software Cost and Performance Measurement," International Journal of Software Computing and Testing, vol. 8, no. 1, pp. 6-12, 2022.

[24] T. Sharma, A. Bhardwaj, and A. Sharma, "A Comparative study of COCOMO II and Putnam models of Software Cost Estimation," vol, vol. 2, pp. 1-3, 2011.

[25] J. Rashid, S. Kanwal, M. Wasif Nisar, J. Kim, and A. Hussain, "An Artificial Neural Network-Based Model for Effective Software Development Effort Estimation," Computer Systems Science and Engineering, vol. 44, no. 2, 2022.

[26] A. Yadav and A. Sharma, "Function point based estimation of effort and cost in agile software development," in Proceedings of 3rd international conference on internet of things and connected technologies (ICIoTCT), 2018, pp. 26-27.

[27] K. Zhang, X. Wang, J. Ren, and C. Liu, "Efficiency improvement of function point-based software size estimation with deep learning model," IEEE Access, vol. 9, pp. 107124-107136, 2020.

[28] S. A. Butt et al., "A software-based cost estimation technique in scrum using a developer's expertise," Advances in Engineering Software, vol. 171, p. 103159, 2022.

[29] M. Dashti, T. J. Gandomani, D. H. Adeh, H. Zulzalil, and A. B. M. Sultan, "LEMABE: a novel framework to improve analogy-based software cost estimation

using learnable evolution model," PeerJ Computer Science, vol. 8, p. e800, 2022.

[30] A. A. Abdulmajeed, M. A. Al-Jawaherry, and T. M. Tawfeeq, "Predict the required cost to develop Software Engineering projects by Using Machine Learning," in Journal of Physics: Conference Series, 2021, vol. 1897, no. 1: IOP Publishing, p. 012029.

[31] A. Karimi and T. J. Gandomani, "Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm," International Journal of Electrical and Computer Engineering, vol. 11, no. 1, p. 707, 2021.

[32] P. V. AG, A. K. K, and V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach," Electronics, vol. 10, no. 10, p. 1195, 2021.

[33] M. Ullah, R. Ali, M. Ahmad, T. Khan, and F. U. Mulk, "Software cost estimation–a comparative study of COCOMO-II and Bailey-Basili Models," in 2019 International Conference on Advances in the Emerging Computing Technologies (AECT), 2020: IEEE, pp. 1-5.

[34] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," Neurocomputing, vol. 385, pp. 100-110, 2020.

[35] M. M. Al Asheeri and M. Hammad, "Machine learning models for software cost estimation," in 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), 2019: IEEE, pp. 1-6.

[36] H. A. Mojeed, A. O. Bajeh, A. O. Balogun, and H. Adeleke, "Memetic approach for multi-objective overtime planning in software engineering projects," 2019.

[37] A. Ullah, B. Wang, J. Sheng, J. Long, M. Asim, and F. Riaz, "A Novel Technique of Software Cost Estimation Using Flower Pollination Algorithm," in 2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS), 2019: IEEE, pp. 654-658.

[38] S. Jha, R. Kumar, M. Abdel-Basset, I. Priyadarshini, R. Sharma, and H. V. Long, "Deep learning approach for software maintainability metrics prediction," Ieee Access, vol. 7, pp. 61840-61855, 2019.

[39] R. Saljoughinejad and V. Khatibi, "A new optimized hybrid model based on COCOMO to increase the accuracy of software cost estimation," Journal of Advances in Computer Engineering and Technology, vol. 4, no. 1, pp. 27-40, 2018.

[40] A. Puspaningrum and R. Sarno, "A hybrid cuckoo optimization and harmony search algorithm for software cost estimation," Procedia Computer Science, vol. 124, pp. 461-469, 2017.

[41] E. E. Miandoab and F. S. Gharehchopogh, "A novel hybrid algorithm for software cost estimation based on cuckoo optimization and k-nearest neighbors algorithms," Engineering, Technology & Applied Science Research, vol. 6, no. 3, pp. 1018-1022, 2016.

[42] H. B. Yadav and D. K. Yadav, "A fuzzy logic based approach for phase-wise software defects prediction using software metrics," Information and Software Technology, vol. 63, pp. 44-57, 2015.

[43] F. S. Gharehchopogh, I. Maleki, and A. Talebi, "Using hybrid model of artificial bee colony and genetic algorithms in software cost estimation," in 2015 9th International Conference on Application of Information and Communication Technologies (AICT), 2015: IEEE, pp. 102-106.

[44] Z. A. Dizaji and F. S. Gharehchopogh, "A hybrid of ant colony optimization and chaos optimization algorithms approach for software cost estimation," Indian Journal of science and technology, vol. 8, no. 2, p. 128, 2015.

[45] W. L. Du, L. F. Capretz, A. B. Nassif, and D. Ho, "A hybrid intelligent model for software cost estimation," arXiv preprint arXiv:1512.00306, 2015.

[46] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," IEEE Transactions on Software Engineering, vol. 40, no. 6, pp. 603-616, 2014.

[47] I. Attarzadeh and S. H. Ow, "A novel algorithmic cost estimation model based on soft computing technique," Journal of computer science, vol. 6, no. 2, p. 117, 2010.

[48] I. Attarzadeh and S. H. Ow, "Improving the accuracy of software cost estimation model based on a new fuzzy logic model," World applied sciences Journal, vol. 8, no. 2, pp. 177-184, 2010.

[49] S. Malathi and S. Sridhar, "A classical fuzzy approach for software effort estimation on machine learning technique," arXiv preprint arXiv:1112.3877, 2011.

[50] T. O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not," Geoscientific Model Development, vol. 15, no. 14, pp. 5481-5487, 2022.